5-11-2013

A Method for Network Clustering and Cluster Based Routing

Jonathan Adelson
University of Connecticut, jonadelson@gmail.com

A Method for Network Clustering and Cluster Based Routing

Jonathan Adelson

BA, Transylvania University, 2001

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

at the

University of Connecticut

2013

APPROVAL PAGE

Master of Science Thesis

A Method for Network Clustering and Cluster Based Routing

Presented by

Jonathan Adelson, BA

Major Advisor _____
Dr. Jun-Hong Cui

Associate Advisor _____
Dr. Reda Ammar

Associate Advisor _____
Dr. Zheng Peng

University of Connecticut

2013

The author would like to acknowledge the assistance of several people without whom this project would not have been possible. First, Dr. Jun-Hong Cui, who's guidance led to the problems studied and made the solutions discoverable. Next, Yan Li, who's work on SACA inspired this work, and whose kind explanations made those solutions achievable. Finally, Dr. Jill L. Adelson whose statistical know-how, love, and support made the entire work possible.

TABLE OF CONTENTS

Chapter 1: Introduction **9**

1.1 Motivation 9

1.2 Topological Group Identification and Overlapped Clustering . . . 12

1.3 Hierarchical Routing 14

1.4 Design Challenges 17

1.5 Contribution 18

Chapter 2: Literature Review **19**

2.1 Clustering 19

2.2 Routing 22

2.2.1 Hierarchical Wireless Routing 22

Chapter 3: Algorithms **27**

3.1 Clustering Algorithm Design 27

3.1.1 Design 27

3.1.2 Notation 29

3.2 SCM-based Clustering with Shared Nodes (SCSN) 30

3.2.1 Overlapping Clusters 31

3.2.2 The Clustering Algorithm 31

3.2.3 Pruning 32

3.3 Routing Algorithm Design 34

3.3.1 Design Goals 34
3.3.2 Network Topology 35
3.4 Scalable Cluster Based Routing(SCBR) 36
3.4.1 The Routing Algorithm 36

Chapter 4: Methodology and Evaluation **39**

4.1 Methodology 39
4.1.1 Clustering Simulation 43
4.1.1.1 SACA Clustering Simulation 44
4.1.2 Group Identification 44
4.1.2.1 Group Identification Metric 45
4.1.2.2 Group Identification Input Graphs 47
4.1.3 Routing Simulation 48
4.2 Clustering Algorithm 49
4.2.1 Clustering Accuracy 49
4.2.2 Cluster Size 53
4.2.3 Cluster Overlap 55
4.2.4 Run-time Analysis 56
4.2.5 Clustering Comparison 58
4.2.6 Group Identification 61
4.3 Routing Algorithm 66
4.3.1 Cluster-Heads 66

4.3.2 Control Messages 68
4.3.3 Data Messages 70
4.3.4 Routing Performance 71
4.3.4.1 SCBR Overlapped Cluster Performance Analysis 71
4.3.4.2 SCSN Performance 72

Chapter 5: Conclusion **74**

5.1 Clustering Algorithm 74
5.2 Routing Algorithm 76
5.3 Future Work 77
5.3.1 Clustering 77
5.3.2 Routing 78

Bibliography **80**

LIST OF FIGURES

1 A simple five vertex graph. 27

2 Five vertex graph clustered by SACA. 28

3 Five vertex graph, clustered with overlapping vertices. 28

4 election messages 38

5 Waxman edge probabilities: $\alpha = 0.6$ 40

6 A 3x3 grid topology 42

7 Regression table for SCM 49

8 SCM variation with graph size - Grid 50

9 SCM variation with graph size - Transit Stub 51

10 SCM variation on grid topologies 52

11 Regression table for cluster size 54

12 Regression table for cluster overlap 55

13 SCM variation on a 1000 node graph 58

14 Mean difference of SCM and Cluster Size between SACA and SCSN 59

15 SCM variation on various graph sizes - Transit Stub 60

16 SCM variation on various graph sizes - Exponential 61

17 AGED variation on graphs of various link densities 62

18 AGED variation on graphs of various link densities. AGED here is not corrected for missing and extra clusters. 63

19 UAGED plotted against node overlap for SCSN 64

20 UAGED plotted against node overlap for SACA 65

21 AGED and UAGED in the data 66

22 Regression table for the number of cluster heads 68

23 Regression table for the number of control messages 69

24 Regression table for the number of data messages 70

25 Normalized cluster-heads between SACA and SCSN clusters . . . 71

26 Mean difference of SCBR Cluster-heads between SACA and SCSN 72

27 Normalized Control Messages and Data Messages 73

List of Algorithms

1 SCSN Clustering Algorithm 33

2 The routing algorithm election process 37

3 The routing algorithm 38

Chapter 1

Introduction

1.1 Motivation

In recent years, two important technologies have changed the way we think about computing, wireless networking, and social networking. Wireless networking has allowed us to connect to the Internet in ways never before possible. While wireless networking has allowed us to connect from the physical world in unprecedented ways, on-line social networking has allowed us to connect with others in the social world in new and exciting ways. While Facebook and Twitter are the current hot social networks, many other websites and companies are tying into social networks or incorporating social network features into their existing products.

Web-based social networks attempt to model the interaction of people using electronic systems. Traditional computer communications tending to connect

users one on one the way telephones typically do. Group based communications historically resemble teleconferencing (i.e. a group phone call) much more than our current social network systems. What differentiates modern social networks from traditional computer communication platforms such as email or instant messaging is the focus on broadcast communication, where a message by default goes out to all of the user's contacts instead of one or a few selected when the message is sent. This results in a model where users communicate and share in groups rather than communicating one on one.

One result of this is that ideas get disseminated across the the social web as users view and re-share ideas from their connections. In older forms, such as Usenet forums, groups formed around topics such as science or computing (in fact, Usenet is famous for its hierarchical names which generally drilled an topic down into greater and greater focus). Social networks, at least the currently successful ones, are built around making the network primary, rather than the topic. While the models differ somewhat in their implementation and focus, the general idea is the same. Twitter focuses more on mass communication and the broadcast message, a system quickly embraced by companies and celebrities. Facebook focuses more on the social aspects, with longer messages and photo sharing. Others, such as LinkedIn, which shares business contacts, have found other niches to fill.

The benefits of social networks come from their ability to leverage the social network graph to solve various problems. LinkedIn leverages the network of

business contacts to facilitate introductions. Using the LinkedIn network, you can find someone to fill a business need who is connected to your existing associates. This provides a level of trust in a contact or contact of a contact because the connection indicates a positive interaction.

Facebook's social network allows people to interact with their Facebook friends, although the connection is much looser than a LinkedIn connection. Since the connection is social, it could imply a strong friendship or a passing acquaintance. Facebook itself uses the social network information to sell advertising and services to their members. Targeted advertising has always been a goal, and Facebook use the information in the social network data they can collect.

For both wireless and social networks, it can be useful to determine information about the network graph from the connectivity. For social networks, the connectivity represents the social connections between people. Facebook friends, LinkedIn contacts, and twitter followers all make up connections in the social network graph. In a wireless network, the connectivity graph represents connections between computers. While the network connections are not explicitly defined by the hardware in a wireless network or the software in a social network, once they are formed, they can be mapped. In both cases, it is useful to study the structures that emerge.

1.2 Topological Group Identification and Overlapped Clustering

One of the basic structures that emerges from the tangle of network connections is a group. Groups exist in different networks for different reasons, but can be identified using a variety of methods.

The most obvious way to determine if a person is part of a particular group is to ask them. Generally, if there is meta-data available, self identification is the easiest way to identify groups. For instance, in a social network like Facebook, users may self identify as members of a group by "liking" a group or page, using the system's mechanisms to join the network's representation of that group. For instance, a hypothetical user name Abram might self identify by joining group such as "Early Learning Center Members (ELC)." His friend Caleb might also join that group, and another group such as "Monday Playgroup (MP)." Now, Facebook knows that both Abram and Caleb are group members of ELC, but only Caleb is in MP.

Meta-data can be an important method for identifying groups. If the information is present, it makes sense to leverage it to identify groups. Facebook has a meta-data system of "likes" which could be used to identify group members indirectly. If Facebook user Kyle likes the "New York Yankees" Facebook page, that datum, combined with other information available, could be used to identify him as a New Yorker, or simply a sports fan. Facebook could use this information (and does) to send Kyle targeted advertising. Additionally, groups that have no

meaning the real world could be identified and leveraged. If a particular user with a certain set of likes clicks on a particular ad, that ad could be shown to other users with the same (or similar) likes. Since on-line advertising typically works by selling volume of clicks, Facebook profits by showing users ads that they are more likely to click. Some statistical analysis would tell them which groups of likes associate well with advertising clicks.

However, if meta-data are not available, another method is required to identify groups. If we have topology information for a social network, based either on system connections, such as Facebook's "Friends," or based on a communication graph, such as in a communication network, we can use that information to identify groups. Graph clustering describes a variety of techniques for doing this in a mathematical graph. Meta-data clustering could be considered a graph clustering technique.

In this work, we will focus on topological clustering, that is, clustering based on the graph's physical structure. For instance, a group of friends would likely be all connected in a social network. While each friend in the group would likely have connections outside the group, each social group would likely be well connected internally and less well connected outside the group. We will call groups of graph nodes of this structure "clusters." Li et. al. [11] describes it as "nodes in the same cluster are highly connected and between clusters they are sparsely connected."

Li [11], Dongen [6], and others present methods for clustering graphs based on topology. An interesting aspect of most of these clustering algorithms is that they

cluster the nodes in such a way that all nodes are placed into exactly one cluster. This is useful for many things, but if we consider our social network examples, it is clear that this is not the only way to cluster a graph. Consider the case of a member of a social network named Caleb. Caleb has a network connection named Abe who attends his school, and he has another connection named Jill who is in his family group. Caleb's family and school are distinct groups, but Caleb is a member of both groups. If we were to graph the connections between these groups, Caleb would have links to the members of his class and the members of his family, but the members of these groups would not necessarily have connections to each other. Jill and Abe are only connected through Caleb. This situation provides the motivation for the first part of this work, which is an method for doing overlapping clustering.

1.3 Hierarchical Routing

While social networks inspire the idea for our overlapped clustering algorithm, wireless networks inspire the idea for the hierarchical routing aspect of this work. In a wireless network, the structure is defined by the communication links. This means that there are no physical boundaries in the network to impose structure. In a wired Ethernet network, all machines are connected to a single physical medium. While in practice there may be network switches, hubs, or repeaters dividing the network, the logical network is all machines connected to a single

medium. To connect two networks, a router is placed between them which connects to both networks. The router is responsible for routing packets between the two networks.

As mentioned, there is no physical structure in a wireless network to provide a routing structure. In a wired network, routing involves transmitting data packets from router to router through the various connected networks until the destination network is reached. Then the router can transmit the data packet to the appropriate end host. In a wireless network, there is no router structure connecting different parts of the network. All of the network nodes are connected through the same media (typically radio, although potentially sound or light) although unlike a typically wired network, not all machines are reachable.

For example, consider a radio network of three nodes, in which two nodes may be out of range of each other but both able to reach the third. While all three communicate over the radio, without some sort of packet routing, the two nodes that are out of range of each other will not be able to communicate.

This sort of situation creates unique problems for wireless networking. In a wired network, the physical wiring and configuration of the network nodes gives the network a routing structure. While a routing algorithm is still used on top of the network to determine how packets are routed, the underlying structure is known. In a wireless network, building a network structure for packet routing is a problem that must be solved in order to utilize the network. While the nodes

could be pre-configured to act as routers based on their placement, this mitigates some of the advantages of using the wireless nodes in the first place.

Consider node placement in a wireless network. First, there must be a reason to use a wireless network in the first place. Wired networks typically have greater reliability in that interference is less likely to disrupt communication. Wireless networks are typically deployed where it is difficult to run wires, or the wires themselves would interfere with the network application. If the nodes are wireless, they can be placed without regard to the difficulties inherent in physically running cables. If the nodes are battery powered then they can be completely disconnected, and even remotely deployed.

For hostile environments this can be important. In a battlefield monitoring environment, nodes could be deployed by airdrop. As long as the network is not partitioned, a routing algorithm that could be built on the fly would allow nodes to send reports wirelessly through a hostile environment. Environments hostile to networks need not be so blatantly hostile, however. Wilderness or industrial environments might be difficult to run wiring, although the node placement may be less difficult. Finally, underwater environments offer their own unique difficulties, but the need to work wirelessly still pervades.

Having an algorithm for group identification, coupled with the challenges imposed by wireless network routing, inspired the second part of this work, which is an algorithm for doing specialized routing within a particular wireless network

using the group identification (overlapped clustering) algorithm as a basis for the wireless routing algorithm.

1.4 Design Challenges

Topological group identification presents unique challenges in that there is no meta-data with which to identify the groups. The goal is to define a group using only the graph structure. We approach this problem as a graph clustering problem.

The second major challenge for group identification is to identify non-unique groups, again, using only the topology. We present a clustering algorithm that not only handles graph clustering, but also handles non-unique groups. The key here is distinguishing the groups only by the connections. The clustering algorithm should identify nodes that are highly connected as belonging to the same cluster, while not constraining them to only one cluster.

Wireless network routing can be difficult in that the network provides no structure from which to manage routing. There is no difference between a regular node and a routing node, and there is no backbone (the ether in Ethernet), even locally, to pass data. In fact, depending on connectivity, every node in the wireless network may be required to route data.

Hierarchical routing requires that a hierarchy be defined which provides some meaningful structure which can be leveraged for routing. Our approach handles this by leveraging the topological features discovered by graph clustering. Well

connected areas are identified as clusters, and adjacent clusters tend to overlap, identifying especially well connected areas. The overlapped areas are not only well connected to themselves, but they are also well connected to nearby clusters.

1.5 Contribution

In this work we present a method for graph clustering allowing for the clusters in the graph to overlap each other. To evaluate the algorithm, we examine the properties that predict its function and how they interact through simulation. We examine whether the clustering algorithm is able to identify overlapped groups and attempt to measure it's success or failure. Since there are few algorithms for identifying overlapped groups using topological clustering, we compare the clustering algorithm to a non-overlapped clustering algorithm.

In addition, we present a hierarchical routing algorithm for a custom wireless sensor network application designed to limit network traffic and preserve batter life for wireless sensors. The routing network takes advantage of the clustering algorithm to perform hierarchical routing and attempts to use take advantage of the overlapped clustering to optimize network usage. Again, simulation is used to study the performance of the routing algorithm and the properties of the network that affect it.

Chapter 2

Literature Review

2.1 Clustering

A variety of work has been done on graph clustering. The Markov Cluster algorithm (MCL) [5] uses flow simulation to calculate clusters in a graph. The idea of this algorithm is that many paths will exist between vertices in a graph that naturally belong in a cluster so the flow will be denser amongst these vertices.

The Least Cluster Change (LCC) [3] clustering algorithm uses a simple clustering method to cluster one hop vertices. It is a specialized algorithm designed for using in Cluster-head Gateway Source Routing (CGSR) [2], described below. This clustering algorithm selects a cluster-head based on either an ID number or the degree of the vertex in question. It then uses a set of rules designed to change the cluster-heads as infrequently as possible, hence the name.

The SCM-based Adaptive Clustering Algorithm (SACA) is a clustering algorithm presented in [12] which uses a clustering accuracy metric the authors

call the Scaled Coverage Measure (SCM) to identify clusters in a graph. SCM is presented in [6] as an unnamed metric for measuring the quality of a clustering. The work presented in this paper is built off the SACA algorithm, so we will go into SACA in some detail. Since SACA is closely tied to SCM, SCM will be presented first. Although SCM is not named by the author of [6], it is referred to as SCM in [12]. For simplicity, we will use the name SCM.

Briefly, we define some terms we use when discussing graphs and networks. For the components of graphs, we will refer to vertices and edges. When discussing networks, we will refer instead to nodes and connections. However, these terms refer to the same things in different context.

The Scaled Coverage Measure (SCM) is a metric for representing the interconnectedness of a cluster compared with the intraconnectivity between that cluster and the nodes that border it. The range of SCM is [0..1], with 1 representing a fully connected cluster with no connections outside the cluster. An SCM of zero indicates that a vertex is clustered by itself (effectively, the vertex is not clustered at all). SCM for a vertex v_i is given by Formula 1:

$$SCM(v_i) = 1 - \frac{|a| + |b|}{|V_{n_i} \cup C_{n_i}|} \tag{1}$$

where a is the set of vertices that are in the cluster with v_i but not connected to v_i in a single hop, b is the set of vertices that are connected to v_i but not clustered with v_i, V_{v_i} is the set of neighbors of vertex v_i, C_{v_i} is the set of vertices

that are clustered with v_i. V_{v_i} is fixed to the graph structure, but C_{v_i} will vary as the clustering progresses.

What the SCM formula tells us is how well the clustering represents vertices that are tied directly to v_i. The clustering for a vertex is clustered only with the neighbors of that vertex. As the fraction in Formula 1, shown by itself in Formula 2 gets larger, the SCM value goes down.

$$\frac{|a| + |b|}{|V_{v_i} \cup C_{v_i}|} \tag{2}$$

That means as the number of vertices in a and b approach the number of neighbors and fellow cluster members of v_i, SCM goes down. a and b represent "bad" matches, that is, cluster members that are not neighbors and neighbors that are not cluster members of v_i.

SCM is useful because it gives a metric that describes the cluster quality at each vertex which we can use in a graph centric, cluster centric, or node centric way. All of the calculations for the SCM of a vertex are local to the cluster and neighbors of the vertex which can be used as in SACA [12] to simplify calculations of the clustering quality.

SACA [12] is a clustering algorithm which attempts to optimize for SCM at each step of the algorithm. Each vertex, processed singly and in random order, attempts to join with its neighbors by calculating the SCM change for joining a cluster and moving to the cluster which will give the largest benefit. A vertex moving to another cluster could additionally trigger a move by a neighbor vertex

to move into or out of the cluster as well. SACA takes advantage of the fact that SCM changes are local so the only vertices affected are the vertex in question, its neighbors, and the members of the associated cluster (see Formula 1). SACA leverages this to make local decisions in a greedy fashion to build clusters. While the final SCM value for the graph may not be optimal, the calculations are simple per vertex and can be run in parallel to some degree, resulting in a good compromise between optimal overall SCM and simplicity.

The common thread among the algorithms described is that they define clusterings to mean graph partitions. This work takes a looser approach and studies the effects of allowing the clusters to overlap. The clustering algorithm described in this paper takes the fundamentals of SACA and the revised clustering definition and builds a new clustering algorithm.

2.2 Routing

Since we will apply the clustering solution to a routing problem, we will now present research which applies to hierarchical routing.

2.2.1 Hierarchical Wireless Routing

Much work has been done on wireless communications and wireless routing ([15], [4], [16], [17], [7], [10], many others). Wireless routing protocols generally focus on ad-hoc and mobile ad-hoc (MANET) routing [4]. Ad-hoc routing means that there are no routers positioned on network borders which handle routing

between networks and there are no wires between nodes and routers that establish the network structure. The routing tasks must be distributed throughout the network; it is the responsibility of the network nodes to ensure that packets arrive at their destinations. Mobility can mean a variety of things from laptop computers appearing and disappearing on the network, to PDAs and phones whose users are physically moving, to fast moving vehicles or slow moving underwater sensor nodes drifting in currents. Various routing protocols address node mobility and the ad-hoc/distributed nature of the network (e.g. [16], [10]).

Wireless routing algorithms are generally divided into proactive and reactive protocols [9] [1]. Proactive routing protocols (e.g. DSDV [15]) build routing tables continuously, regardless of need [9]. In a proactive routing protocol, when a network node is ready to send data, it can consult its pre-existing routing information and send the data. The routing data in a pro-active routing protocol must be periodically updated in order to maintain its accuracy, especially in a MANET routing protocol. If the nodes are in fixed locations, routes may need to be repaired less often. A proactive routing protocol must still be able to repair routes if they fail between maintenance cycles.

Reactive (or on-demand) routing protocols (e.g. AODV [16], DSR [10]) collect routing information as needed instead of continuously, that is, they react to a routing request and build the necessary routes [9]. This is not to say that a reactive routing protocol cannot cache collected information for later use (for instance [16], but that information is not discovered initially or updated except

in response to a routing request. At request time, if the routing information is up-to-date, it can be used as is. Otherwise, it is updated at the time of the new request. Reactive routing is often used in MANET routing because the position of the nodes will change the routes through the network. Reactive routing assumes that stored routes will be quickly invalidated by node movement; these protocols build the routes as needed.

Hierarchical routing protocols in MANETs will tend to be proactive routing protocols([14], [2], [7]). The need to keep the hierarchical structure intact requires that periodic updates be sent out to keep maintain the hierarchy proactively. Building the hierarchical structure on-demand would be prohibitively expensive.

The Hierarchical State Routing protocol (HSR) [14] is based on the ideal of "group mobility," which is to say, groups of nodes in the network move as a unit, and can be grouped in logical groups, as well as physical groups. These groups form the basis of a sub-netting scheme that does backbone routing through a series of key network nodes. Each group in the network may pass packets to the next cluster based on the next cluster-head in the route. HSR relies on its mobility groups for routing, so when these groups are not available, then a different method will be required.

Cluster-head Gateway Switch Routing (CGSR) [2] uses the LCC clustering algorithm (described above) to cluster the wireless network. This clustering produces an interesting overlapped pattern of clusters, which is then used to route data. The nodes shared between clusters are called Gateways in CGSR. The

CGSR routing algorithm functions by sending data alternately through cluster-heads and gateways. All nodes in the cluster are within transmission range of the cluster-head, so every cluster-head can transmit to a gateway. Since the gateway exists in multiple clusters, it can transmit to multiple cluster-heads. As a result, by sending data from cluster-head to gateway and gateway to cluster-head, packets can be transmitted across the network. CGSR uses a specialized version of DSDV routing to transmit data through the network using the cluster-head gateway routes.

Fisheye State Routing (FSR) [13] is a routing technique that is based on the metaphor of a fisheye lens. The fisheye lens has less distortion near the focal point which degrades further away. Applying this idea to routing, the authors developed a link state technique where the LS table is not flooded throughout the entire network when a change is detected, but instead updated periodically. Nearby nodes are sent updates more frequently, and distant nodes are updated less frequently. As in the fisheye metaphor, this results in high route accuracy in the area local to a node (because of the frequent updates) and low accuracy in areas more distant. As the data travels across the network, the route gets more accurate as it approaches the destination. This is implicitly a hierarchical routing protocol because while the routing scheme is technically flat (not hierarchical) there is an implicit structure locally around each node where the routing updates change.

Landmark Ad-hoc Routing (LANMAR) [7], is a MANET routing protocol that uses a two level hierarchy to transmit data through the network. It is a combination of FSR and a backbone routing scheme called Landmark Routing. Landmark routing functions similarly to the group mobility model described in WHIRL in that it uses an assumption that distinct groups will move together and can be treated as a hierarchical unit. A landmark address identifies nodes based on nearby landmarks. The landmark routing aspect will allow a packet to get nearby to its destination by homing in on landmarks in the network. LANMAR combines the landmark routing aspect with fisheye routing so that the landmarks can be used to send data across long distances and the fisheye updates can be used to send data locally.

Zone Routing Protocol (ZRP) [8] divides nodes into routing zones. Nodes on the border of the zones are responsible for providing routes across the network. Routing internal to a zone is handled proactively, while routing across the zones is handled reactively via an on-demand routing protocol. Routing is handled by border nodes through a system called border-cast, which is primarily a multicast across the zone. Border nodes are used like backbone nodes, and routes are created between them. When a node wants to transmit within its zone, it can send data using its prebuilt table. When it the destination is not in the zone, the border nodes provide an on demand route to the appropriate zone, where in-zone routing can take place.

Chapter 3

Algorithms

3.1 Clustering Algorithm Design

3.1.1 Design

The design of the clustering algorithm is motivated by observations while running the SACA clustering algorithm. Consider the network presented in Figure 1. There are numerous possible clusters with only these 5 vertices.

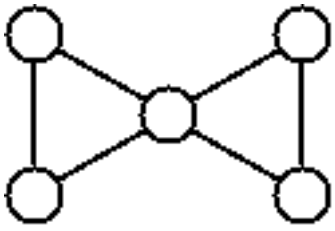

Figure 1: A simple five vertex graph.

The best clustering accuracy as measured by SCM value can be achieved by clustering as in Figure 2. This is the output that the SACA clustering algorithm would produce.

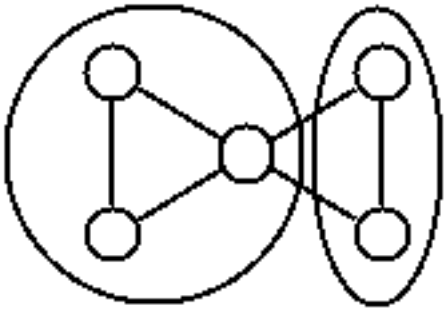

Figure 2: Five vertex graph clustered by SACA.

The clustering accuracy for this clustering is the best SCM value that can be achieved in this graph section. However, the central vertex can work equally well in either cluster. This example suggests that there is another possibility. If we relax the definition of a cluster so that it is not simply a graph partition and allow the vertices to join multiple clusters, another possibility emerges. This is illustrated in Figure 3.

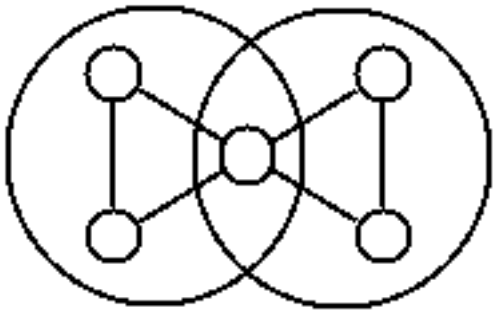

Figure 3: Five vertex graph, clustered with overlapping vertices.

If we use the SCM value (see Formula 1) as the measure of clustering accuracy, there is a clear improvement in the overall SCM values for the graph. In the clustering given by Figure 2, the vertices in the two vertex cluster each have an SCM value of 0.5. Two of the vertices in the three vertex cluster have SCM values

of 1, and the central vertex has an SCM value of 0.5. The average SCM value of the vertices in the graph is therefore 0.7.

In the graph described by Figure 3 the SCM values are improved. All four perimeter vertices have an SCM value of 1, because they are fully clustered with their neighbors. However, this presents an interesting problem. The central vertex in the graph now has two SCM values, one for each cluster. For each separate cluster, the shared node will have an SCM value of 0.5. If we combine these SCM values by averaging, then average the SCM values of the entire graph, we calculate an overall SCM value of 0.9. This is a clear improvement in SCM value so we will explore other properties of this method of clustering.

While in this example it seems natural which vertices should be clustered together and which vertices should be shared among clusters, in more complex graphs it may not be obvious. Part of the results of this work will be to study the consequences of this clustering scheme on more complex graphs.

3.1.2 Notation

The clustering algorithm will operate on an undirected graph G. Let graph $G = (V, E)$ be a connected graph in which V is the set of vertices in the graph and E is the set of edges. In this paper, a cluster is a subset of V with a high concentration of internal edges. The set of i clusters, the union of whose vertices make up G, is denoted $C = \{C_0, C_1, C_2, ...C_i\}$. Recall that in this clustering

algorithm, the intersection set of the vertices of the clusters in C need not be empty, that is, the clusters in C can share vertices.

Every vertex in the graph has an SCM value. The SCM value for a particular vertex is calculated by the formula for SCM, presented previously. This SCM value represents the quality of the current cluster in which the vertex is involved. The SCM for a cluster, c_x, is the average of all the SCM values for the vertices included in c_x.

Since the clusters can share vertices, each vertex will need an SCM value for each of its containing clusters. This means that vertices will have up to $|V| - 1$ SCM values, potentially. Vertices will have to keep track of the SCM values for all of their containing clusters, but at times will operate using the averaged value. As in [12], we will refer to the graph SCM as the average SCM of all vertices in the graph.

3.2 SCM-based Clustering with Shared Nodes (SCSN)

We now present a new clustering method based on the idea of relaxing the usual clustering definition. Typically, a cluster is one partition resulting from partitioning a graph, thus, the vertices in a particular cluster are unique to that cluster. SCM-based Clustering with Shared Nodes (SCSN) is a relaxation of the clustering definition to allow vertices to be shared among clusters. It is based on the basic premise of SACA in that it uses the clustering accuracy metric (SCM)

to decide which vertices join a particular cluster, with the goal of an overall high graph average SCM value.

3.2.1 Overlapping Clusters

This relaxation has certain consequences evident in the resulting clustering. First, there is no longer an intuitive clustering for all graph patterns. A particular vertex could be unique to a particular cluster, shared among multiple clusters, or (potentially) alone. In practice, no vertex will ever be left out of a clustering because a higher SCM value can always be achieved by connecting to any one-hop neighbor vertex.

Second, there can be two identical clusters formed separately with the same included vertices. Because two clusters can share vertices, it is impossible to tell if they will end up with the same members until the clustering is complete. Similarly, there could be two clusters that have nearly identical membership with a few differences. These cases may need to be handled differently depending on the application in question.

3.2.2 The Clustering Algorithm

The SCSN clustering algorithm works as follows. Each vertex is processed individually. Initially, a vertex is considered to be a cluster by itself. The current cluster adds (one at a time) any neighboring vertices which will result in a positive SCM gain for the vertices that are already in the cluster (that is, a positive average

SCM benefit for the cluster; a particular vertex in the cluster could experience an SCM decrease). The vertices are added to the cluster in order of SCM benefit. See Algorithm 1.

The algorithm treats the initial vertex as a cluster of one, and adds other vertices to that cluster picked from the cluster's neighbors. If there is no benefit to the cluster to adding a particular vertex, that vertex will not be added. A particular vertex that is not added in one round may be added in a later round depending on the vertices that already exist in the cluster. Once the vertex has joined the cluster, it participates in SCM value calculations and its SCM value is included as other vertices attempt to join the resulting cluster.

As vertices join clusters, some clusters will become subsets of other clusters. These clusters are collapsed into a single cluster such that all proper subsets are discarded. If the intersection of two clusters are not equal to one of the clusters, the two clusters are not joined. This is discussed further in section 3.2.3.

3.2.3 Pruning

The algorithm adds vertices to each cluster starting from a particular vertex. The cluster grown from a given vertex may be distinct from all other clusters or the same two clusters could be grown from two different starting vertices. With a group of clusters, if any clusters are subsets of the others, they may be discarded. However, we can detect this earlier in the process.

Algorithm 1 SCSN Clustering Algorithm

```
for all vertices v_i in G(V, E) do
  while adding new vertices do
    c ← v_i
    SCM_max ← 0
    for vertices v_j incident on c in random order do
      calculate ΔSCM for v_j joining c
      if ΔSCM > SCM_max then
        v_max ← v_j
        SCM_max ← ΔSCM
      end if
    end for
    if SCM_max > 0 then
      c ← v_max ∪ c
    end if
  end while
  for cluster c_i where v_i ∈ c_i do
    if c ⊂ c_i then
      discard cluster c
    end if
  end for
end for
```

A clustering is dependent on a starting cluster and the topology. For example, a vertex 'A' will grow a particular clustering based on its connections. The only randomness is if two vertices are available with the same SCM benefit. In this case, one will be chosen randomly to be added. If we disregard this bit of randomness, we can prune the decision tree by stopping and discarding a cluster if the current clustering has previously occurred. That is, if a particular clustering has already been built starting from a particular vertex, we can assume that it will be build again the same if that clustering appears built from a different vertex. The order in which a cluster is built does not affect the future clustering.

This is best illustrated by an example. Suppose that vertex A builds a cluster by adding B, then C, then D. The cluster states are A, then AB, then ABC, then ABCD. Now also suppose that vertex B grows a cluster by first adding vertex A. Since we've already calculated that cluster AB grows into cluster ABCD, we can stop growing B's cluster and use A's cluster instead. This is trivial in a centralized implementation, but would require some message passing in a distributed implementation.

3.3 Routing Algorithm Design

3.3.1 Design Goals

The goal of the routing algorithm is to leverage the SCSN clustering algorithm to produce a routing algorithm for specialized wireless networks. The routing algorithm should support a network of freely distributed stationary nodes with

limited power. The primary goal is for the network to be self organizing with an emphasis on limiting the number of required network transmissions and distributing the transmission load across the network as well as possible. Nodes that have infrastructure or repeater tasks to perform should hand off those tasks such that the additional transmission load is shared as well as possible through the network.

The SCSN clustering algorithm should lend itself well to developing this routing algorithm because the shared nodes in the clustering provide useful information about how the nodes are connected and can share duties transmitting data in the network.

3.3.2 Network Topology

The network consists of a collection of sensor nodes which collect data and transmit it to a base station. The nodes are distributed randomly on a plane in the x and y directions. All of the nodes have two transmission levels, a high transmission power level which can reach the base station from anywhere in the network, and a low power level which can reach other nodes in the network. The network is assumed to be connected when the nodes are in low power mode, that is, all nodes are reachable through a series of hops.

The nodes in the network describe a general graph, where the nodes in the network are vertices in the graph. The transmission range is the same for each node. Two nodes which are within each others transmission range form a link

by which they can communicate. These links are edges in the graph, and are bi-directional since all nodes have equal transmission range. Network interference is not considered.

3.4 Scalable Cluster Based Routing(SCBR)

3.4.1 The Routing Algorithm

The Scalable Cluster Based Routing (SCBR) algorithm is designed to take advantage of the cluster structure in the graph to perform routing. The clustered nodes elect cluster-heads, which will forward data. Since clusters share nodes, there will be many eligible nodes to be cluster-heads for multiple clusters. In this way, we can maximize the number of nodes that are used as cluster-heads and maximize the number of nodes they serve.

One of the primary goals of the routing algorithm is to distribute the transmission load to multiple nodes. SCBR accomplishes this goal by rotating cluster-heads as the algorithm cycles. Setting a parameter to restrict how often a node can serve as a cluster-head will effectively prevent a node from being overused, within limits imposed by the cluster structure.

The routing algorithm functions as follows. First, the nodes are clustered using the SCSN clustering algorithm. After the cluster structure is established, the nodes perform an election. The election establishes cluster-heads. The cluster-heads may head multiple clusters, as they are more likely to be picked from nodes shared amongst multiple clusters, but they can also only serve a single cluster.

Since SCSN will always cluster a node with another node in a connected graph, no node will ever be serving only itself.

The key to the SCBR routing algorithm is the cluster-head election process. The cluster-heads must be elected, with priority given to shared nodes so that those nodes can serve as many nodes as possible, while distributing the load around multiple nodes over time. This may result in an increase in the number of cluster-heads over time as the algorithm cycles, because in the first election, ideally, the "best" cluster-heads will be picked. In subsequent elections, less optimal cluster-heads will be elected in order to distribute the communication load. Since these cluster-heads are less optimal, they will serve less nodes each and more cluster-heads will be required to serve all of the nodes.

Algorithm 2 The routing algorithm election process

```
e_l ← calculate local election value
e_c ← e_l
send(e_c)
if new election message then
  e_n ← recv(election record)
  if e_n > e_c then
    e_c ← e_n
    next hop ← sending node
    send(e_c)
  else
    resend recv'd election data
  end if
end if
```

Election messages consist of five fields as shown in Figure 4. The *counter* keeps track of the current election. Any messages from a previous election lose a current election. *clusters* is the list of clusters to which this message applies.

counter	clusters	last_hop	hop_ct

Figure 4: election messages

If a node is not in any of the clusters in *clusters*, the message is discarded. The *last_hop* is the id of the sending node. The *hop_ct* is the number of hops to the nearest cluster-head. When comparing two election message, a node first compares the election counter, then checks to see if it is in any of the clusters listed, then compares the last_hop id with its current best id and selects the greatest value.

The routing algorithm continues until the nodes are sufficiently used up that the graph is partitioned or all of the nodes have run out of battery. Any nodes that are disconnected from the graph have no need for routing. This gives us Algorithm 3.

Algorithm 3 The routing algorithm

```
repeat
  if node death then
    re-cluster
    run election
  end if
  while data message generated or received do
    if cluster-head then
      sendto(base station, data)
    else
      sendto(next hop, data)
    end if
    sendto(next hop, data)
  end while
until all nodes depowered
```

Chapter 4

Methodology and Evaluation

Here we will describe the methodology and tools used to evaluate the algorithms presented in Chapter 3. In addition, we will describe the results of the simulations and provide an analysis of them.

4.1 Methodology

In order to study the algorithms, we wrote two simulations, a clustering simulation and a routing simulation. The simulations will be described in detail below. In addition to the algorithm simulations, we wrote graph generators to generate wireless network type disk graphs and grid type graphs.

Using the BRITE topology generator, we generated random topologies, which we used as a baseline for testing. Random topologies have a certain probability that any two nodes will be connected. If the probability is 0.4, nodes will be connected 40% of the time. Random topologies have no structure, so they make

a good base graph type for testing. Each other topology type (except grid) introduces structure but preserves a certain amount of randomness at the same time.

Waxman topologies are commonly used for testing, although they don't generally give realistic network topologies. Waxman topologies are generated using a probability function with configurable parameters which affect how edges are created. The probability function is based on the relative distances between vertices in the graph and has two parameters, α and β.

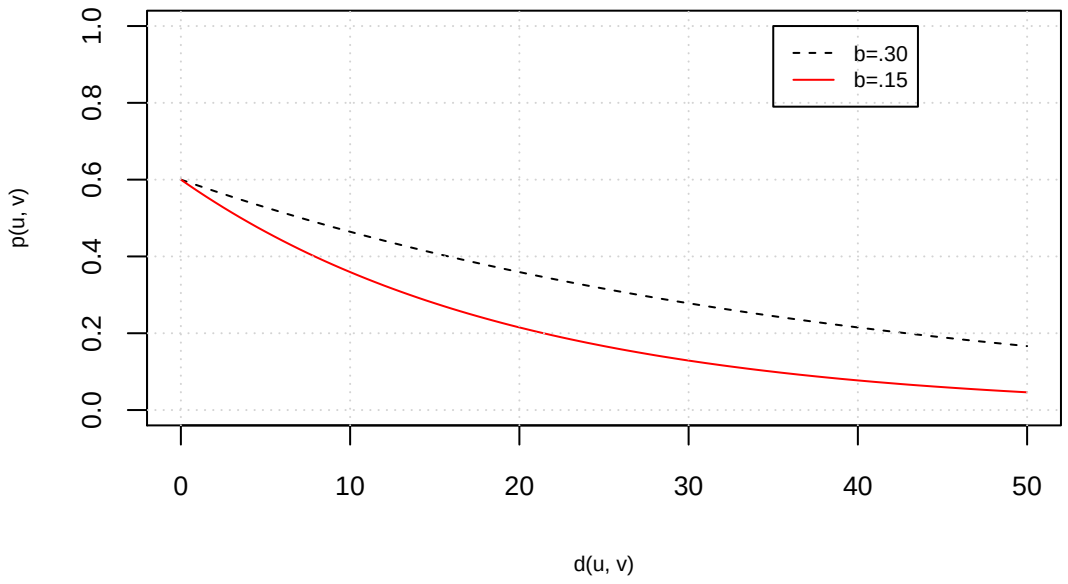

Figure 5: Waxman edge probabilities: $\alpha = 0.6$

Figure 5 shows the variation in edge probability between vertices (u, v) as the distance between the vertices increases. The horizontal axis is the distance

between vertices (u, v) and the vertical axis is the probability of a edge. In the figure, α has been set to 0.6, which is the maximum value in both curves. α is similar to the probability in a random graph, but reduced by the β parameter. In the upper, dashed curve, the value for β has been set to 0.3. In the lower, solid curve, the value for β has been set to 0.15. It is clear that with a lower value for β, the distance has a greater effect on the probability for a edge.

Transit-Stub graphs are designed to match topologies that exist in real networks in many ways. Transit-Stub topologies are built in a hierarchical fashion to mimic the way real networks are designed, with each level having random graphs as their basis. The result is a graph that has connections at multiple levels and reflects real world designs.

Exponential topologies are topologies in which a small number of nodes have many connections and a large number of nodes have a very few connections. Exponential topologies have properties that are similar to properties in real networks.

Two specialized topologies were generated for the purposes of testing the clustering and routing algorithms presented here. One is a grid topology, where all nodes are distributed in rows and columns and edges connect nodes to the north, east, south and west. The resulting graph describes a grid layout. While this graph is not realistic as a network graph, it does provide an interesting perspective on the clustering algorithm. We generated grid graphs using a custom python script. See figure 6.

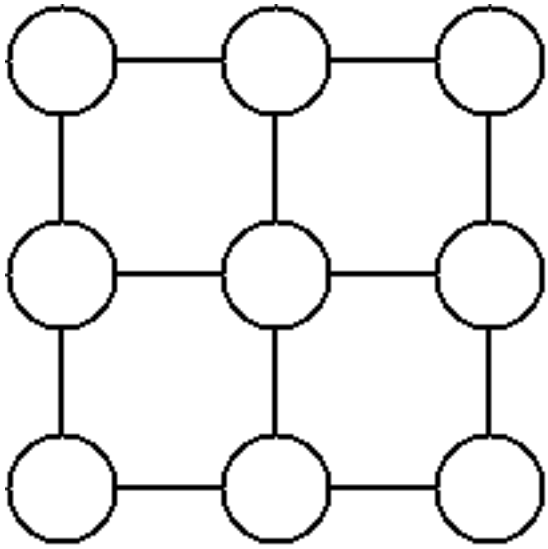

Figure 6: A 3x3 grid topology

Additionally, we created a wireless-range type graph which is generated as follows. Nodes are distributed randomly in a plane. The nodes are then connected using the distance formula with a fixed maximum connection range. This is to simulate wireless nodes distributed in a relatively clear area where the only limit to connectivity is transmission range.

The routing simulation uses a graph clustering to execute the routing algorithm. The clustering could theoretically come from any source that provided overlapped clusters, but for the tests performed here they come exclusively from the clustering simulation. Therefore, before running the routing simulation on a graph, we first run the clustering simulation to establish a clustering. Then the routing simulation uses the graph and the output of the clustering simulation to perform routing. For comparison purposes, we applied the routing algorithm to a SACA clustering as well as to an SCSN clustering.

We generated multiple instances of each network topology for input into the algorithm simulator. Additionally, we ran the simulations of each algorithm repeatedly to give us enough data to study the randomness that can occur within the algorithms. This allows us to examine what properties of the algorithms and of the topologies affect the outcomes.

4.1.1 Clustering Simulation

The clustering simulation takes as input a graph describing vertices and the edges between them in the form of an edge list. The simulation functions by stepping through each vertex in the graph and applying the algorithmic rules to create a cluster. Once a particular cluster has grown to completion, the cluster is compared with existing clusters to see if it is a subset or super-set of an existing cluster. After any proper subset clusters have been discarded, the algorithm is free to continue clustering further cluster. See the algorithm details 1 for more information.

This algorithm is sensitive to the order in which nodes are clustered, in a manner similar to SACA. For the SCSN algorithm, if two nodes have equal SCM benefits to the graph, one is selected at random to be added to the cluster. By collecting data on multiple runs of the algorithm over the same data set (input graph), we will be able to measure the variation caused by this randomness.

The clustering simulation records various information on the resulting clustering, including SCM values for all nodes, SCM value for the complete graph

clustering, and the clusters produced. Additionally, the simulation script provides the average cluster size and the total number of clusters produced, as well as a measure of the amount of overlap in the clusters.

4.1.1.1 SACA Clustering Simulation

For comparison, we used a clustering simulation provided by the authors of [12]. This clustering simulation uses the SACA clustering algorithm to cluster the network nodes. The SACA clustering simulation takes similar inputs, including a network topology, and produces a clustering and some statistics on the results of the clustering. In any case where the SACA algorithm fails to cluster the graph the program is re-run to ensure sufficient comparison data.

4.1.2 Group Identification

The clustering algorithm is designed for group identification. Beyond simply collecting data on the properties of the clustering algorithm, we collect data to examine the ability of the algorithm to identify groups within the network graph. We coded a special topology generator to examine the group identification problem. This graph generator is based on the idea that if we start with a series of groups, then allow them to interact in some way, we should be able to recall the original groups using only the graph topology.

The graph generator works as follows. It begins with a series of groups. The groups are assumed to be well connected internally. Then, the generator equates

a selection of nodes such that two nodes from different groups become a single node, taking on all of the links of both original nodes. As an example, consider a student in a college class. That student knows everyone in her class. However, the student is in multiple classes. If the student is in classes A and B, that student will have links to all of the other students in both classes, although the other students in A may not have links to the students in B. The link structure suggests to the clustering algorithm that the student in question, well connected to the students in both classes, should belong to both groups.

4.1.2.1 Group Identification Metric

For evaluation of the group identification results, we developed a metric based on the Levenshtein distance between sets. Normally, the Levenshtein distance, or edit distance as it is commonly known, is applied to strings. Basically, it is the amount of edits (insertions, deletions, alterations, transpositions) required to change one string into another. This is typically used in spell checkers, although it has many other uses. In this case, we are using the edit distance between sets as our metric for group identification accuracy.

Consider the generated clusters and the original clusters to be sets. We use the edit distance between the sets to determine how well we generated the clusters from the topology data as compared to the clusters we started with. If we recreate a cluster exactly, the edit distance is zero. For every extra or missing node in the generated cluster, one edit is counted. A wrong node in the cluster also counts

as one edit. Unlike string edit distance, our sets are unordered, so we do not consider transposition.

Since the clusters may not match exactly, we use the edit distances to pair up the generated clusters with the original clusters. The pairs with the closest edit distance are matched. In the case of ties, the lowest total edit distance is considered across all of the clusters. If there is still a tie, a random match is chosen.

To evaluate a single cluster, the edit distance is used, but normalized against the size of the cluster. Here, let function $ed(c)$ represent the edit distance between the cluster c and the original group it represents. If cluster c has size $|c|$, then the Normalized Edit Distance (NED), is:

$$NormalizedEditDistance = ed(c)/|c| \tag{3}$$

The Normalized Edit Distance ranges from zero to one. In a large cluster, a single missed node will not be as large of a penalty to the NED as in a small cluster. That is, if a three node cluster is missing a node, the NED will be larger (0.33) than if a ten node cluster is missing a node (0.1). A missing cluster has an NED of 1, as does an extra cluster.

The metric used for evaluating the quality of the clusters is as follows. The NED is summed for all clusters. For n generated clusters $c_0, c_1, ..., c_n$, the equation:

$$AvgGraphEditDist = \frac{\sum_{i=0}^{n}(ed(c_i)/|c_i|) + m + e}{n} \tag{4}$$

gives us the average graph edit distance (AGED), where m is the number of missing clusters and e is the number of extra clusters, that is, m is the number of missing clusters times an NED of one for each. The same holds for e.

While the NED ranges from zero to one, the AGED can go beyond one. Extra clusters that are detected can skew the metric, because an extra cluster adds one to the numerator, but leaves the denominator unchanged. This does not happen for missing clusters because they vary between zero and n, that is, the denominator. Regardless, this gives us a metric for the graph to indicate how well the clusters match the original set of groups, in a similar way that SCM gives us information about the quality of the clustering as compared to the network topology.

4.1.2.2 Group Identification Input Graphs

In order for group identification to be possible, starting groups need to be known. After the clustering is complete, the edit distance is calculated between the starting groups and the clusters. So the graphs based on the starting groups are built as follows.

First, groups are generated. Then, nodes are randomly picked between groups to be "equated," that is, two nodes are selected, then replaced with a single node which is end point to all of the links to which the two selected nodes were endpoints. This reduces the number of nodes in the graph, but results in a graph of overlapped clusters. When all links are present, all groups are fully connected.

The graphs are generated with a percentage of links removed so that groups are not fully connected. Fully connected groups should be easier to detect because they are better defined.

4.1.3 Routing Simulation

The routing simulation is designed to compare the routing algorithm using the SCSN (overlapped) clustering with a traditional clustering. The traditional clustering is created using the SACA clustering algorithm. The routing algorithm, while designed for use with an overlapped clustering, will function with clusters that do not overlap. Therefore, the routing algorithm is run on the same graphs with both the overlapped and non-overlapped clustering. Our analysis will consider if the overlapped clustering provides a benefit to the routing algorithm.

The routing simulation takes a clustering and the associated graph and simulates the SCBR routing algorithm. The primary goal of the routing simulation is to count the number of routing messages required to perform the routing algorithm and the number of cluster heads that will be established. In order to measure the performance of the algorithm properly, the simulation iterates the algorithm over time. Thus new routes can be calculated as the routing algorithm moves forward.

4.2 Clustering Algorithm

In this chapter we present the analysis of the algorithms described in Chapter 3. The experimental methodology is described in Chapter 4.

4.2.1 Clustering Accuracy

The Scaled Coverage Measure (SCM) is our measure of clustering accuracy. First we will look at graph attributes which predict clustering accuracy.

A regression analysis (Figure 7 shows us what predictors contribute to the SCM value in particular clustering. The regression predictors include the number of vertices in the graph, the vertex degree, and the type of topology. The regression model explains a large proportion of variance such that adjusted $R^2 = 0.986$. The number of vertices (n) in the graph did not have a practically significant effect on clustering accuracy, that is, $b = -2.07e^{-06}$. This means that when adding 1000 vertices to the graph, one could only expect a .002 decrease in the SCM of the graph.

predictor	b-value	std. error	p-value
n	$-2.07e^{-06}$	$5.84e^{-07}$	$< .001$
degree	$-2.06e^{-02}$	$1.76e^{-04}$	$< .001$
bWaxman	$3.69e^{-02}$	$8.93e^{-04}$	$< .001$
bTS	$2.79e^{-01}$	$1.06e^{-03}$	$< .001$
bGrid	$7.90e^{-02}$	$1.82e^{-03}$	$< .001$
bExp	$2.55e^{-01}$	$1.13e^{-03}$	$< .001$

Figure 7: Regression table for SCM

The average node degree of the graph was more strongly (negatively) predictive of clustering accuracy, where $b = -0.02$. This makes sense in that the connections in the graph are what make the clusters, regardless of the number of vertices. The clustering is entirely local, as discussed in the algorithm design.

Here we will again attempt to show the effect of graph size on the SCM value. The regression has shown that n is a poor predictor of SCM value. Figure 8 shows the SCM value variation on a series of sizes of grid topologies. Note the scale on the Y-axis. On very small graphs, there is slightly more SCM variation, but this decreases as the graph size increases.

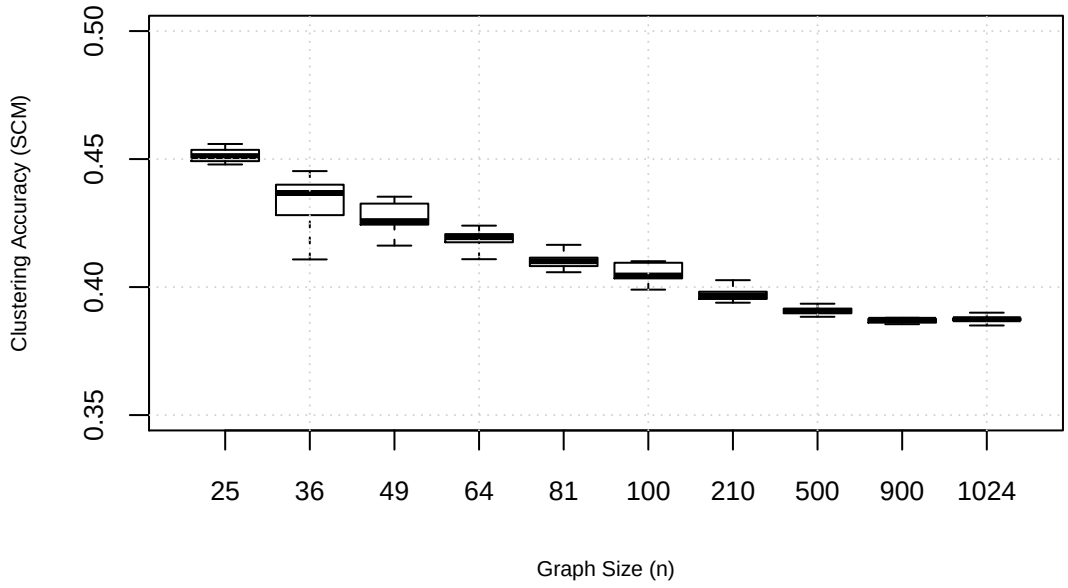

Figure 8: SCM variation with graph size - Grid

Figure 9 shows the SCM variation on various sizes of Transit-Stub graphs. Again, note the scale on the Y-axis. The SCM variation is quite limited as affected by n.

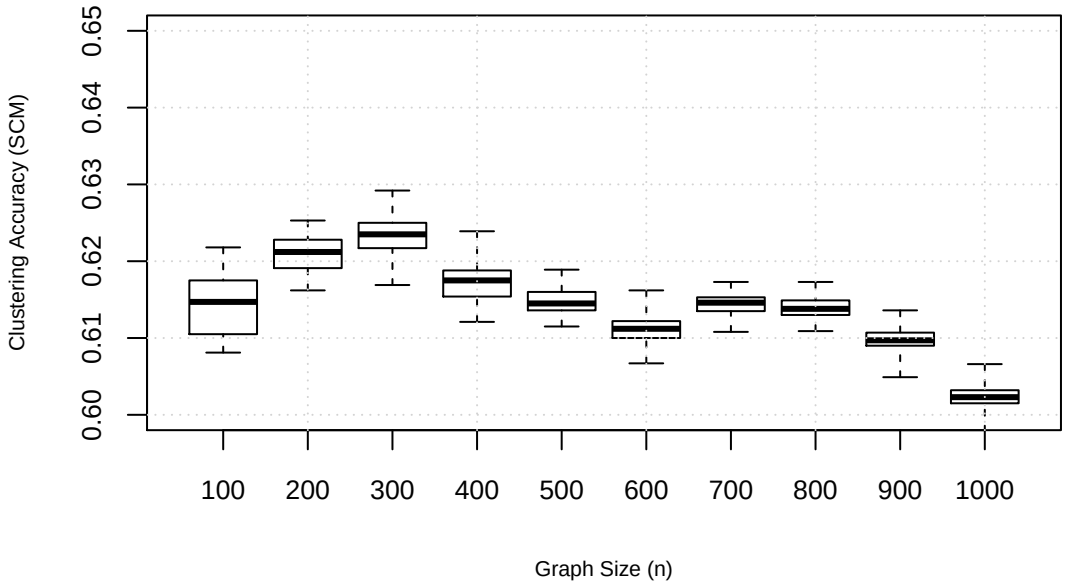

Figure 9: SCM variation with graph size - Transit Stub

For the regression model, the topologies were dummy coded with the pure random topology as the reference category. The other topologies tested were Waxman, transit-stub, grid, and exponential topologies. When talking about the b-values of the topology types, the values represent the difference from the random topology type. For Waxman topologies, $b = 0.037$, the smallest b-value for the topology types. This makes sense as the Waxman topologies are similar to pure random topologies with a distance parameter taken into account.

For grid topologies, $b = 0.08$. Grid topologies are not realistic, and the degree and SCM values are relatively static. The only change to the average vertex degree in a grid topology is a result of the ratio of internal vertices (degree four) to edge vertices (degree three). All grid topologies will have a degree between 2 and 4 (note that the corner vertices of the grid are of degree two). SCM values given by grid topologies are also relatively static when clustered by SCSN. See Figure 10.

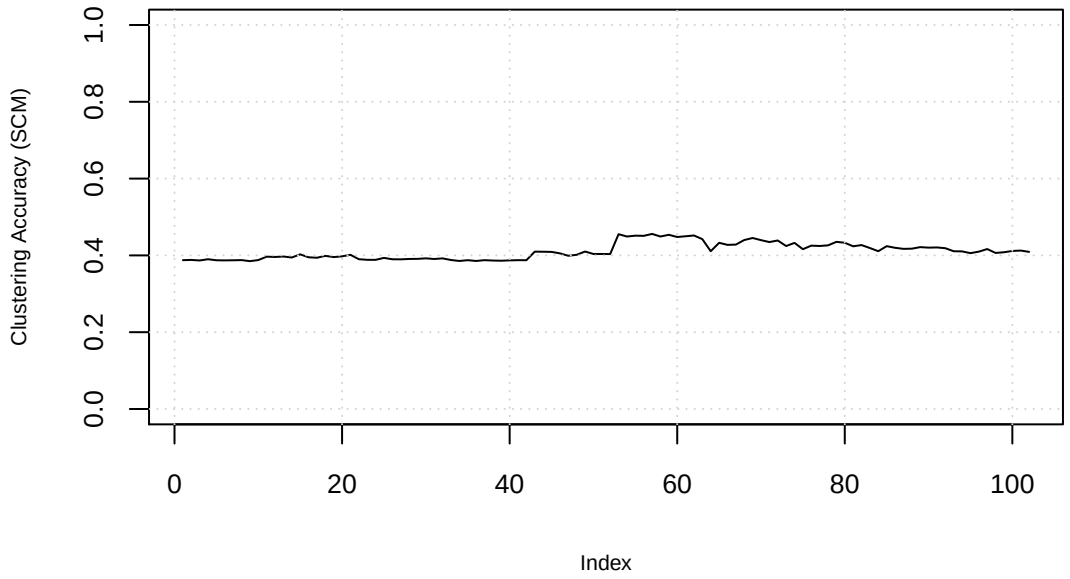

Figure 10: SCM variation on grid topologies

Transit-stub topologies give more realistic networks, and they are built in a hierarchical fashion. These network emphasize local connectivity and are therefore naturally amenable to clustering. This is born out with a relatively high

b-value, that is, $b = 0.28$. This means that being a Transit-Stub topology and controlling for other predictors, a topology will have an SCM value almost .3 higher!

Interestingly, exponential topologies have a similar b value, $b = 0.255$. This indicates that the exponential topologies cluster well as measured by SCM value relative to the random topology type.

There are a few things to note for all topology analysis. First, all p-values are less than 0.001, so all b-values are statistically significant, even if they are too small to be practically significant. Next, while the regression accounts for average degree, it is possible that the clustering is sensitive to parameters of the various topology types. While an effort was made to have some representation of various topology parameters, there are literally infinite possibilities. Given more variation, it is possible that the b-values would change.

It is also interesting to note that all topology types had coefficients (b-values) greater than zero. This means that all topology types are likely to have a greater clustering accuracy than pure random topologies, the reference category. This is intuitive, as all topology types add varying levels of structure to pure random topologies. The result of structure is improved clustering characteristics.

4.2.2 Cluster Size

The size of the clusters is the next thing we will discuss. The regression model (Figure 11) again explains almost all of the variance, that is, adjusted $R^2 = 0.970$.

As before, the number of nodes in the graph had a minimal impact on the cluster size, $b = -1.37e^{-04}$. So with an increase of 1000 nodes in the graph, one could only expect a 0.137 node decrease in the average cluster size.

predictor	b-value	std. error	p-value
n	$-1.37e^{-04}$	$4.32e^{-06}$	$< .001$
degree	$2.23e^{-01}$	$1.30e^{-03}$	$< .001$
bWaxman	$-3.14e^{-01}$	$6.61e^{-03}$	$< .001$
bTS	$4.91e^{-02}$	$7.87e^{-03}$	$< .001$
bGrid	$5.87e^{-01}$	$1.35e^{-02}$	$< .001$
bExp	$-7.69e^{-01}$	$8.38e^{-03}$	$< .001$

Figure 11: Regression table for cluster size

Node degree is a decent predictor of cluster size ($b = 0.223$). As mentioned previously, degree is a negative predictor of SCM. As degree goes up, the clusters get bigger and the SCM goes down. This implies that the clusters become multi-hop. Nodes that are clustered but are not directly connected suffer a penalty to SCM.

For the topology types in this regression model, the random topology is still the reference category. None of the topology types have a extremely strong impact on the cluster size. Being a grid topology seems to have the greatest effect with a b-value of 0.587. This indicates that being a grid topology adds half a node to the average size over being a random topology.

The exponential topologies have the greatest effect in the other direction. Being an exponential topology has $b = -0.769$. This is likely explainable by the long tail of low degree nodes an exponential topology. Unless there are

many clusters around the few high degree nodes which are clustered with more members, the long tail in node degree will bring create a long tail of small clusters and bring down the average.

Neither the transit-stub topology ($b = 0.049$) or the Waxman topology ($b = -0.314$) have much practical effect on the cluster size, although Waxman does seem to result in slightly smaller clusters than random.

4.2.3 Cluster Overlap

Cluster overlap is the amount to which the clusters share nodes. In the regression model with an outcome of cluster overlap (figure 12, the model again explains a significant portion of variance ($R^2 = 0.968$). In this case, for the first time, the number of nodes in the graph did in fact have a meaningful effect, $b = 0.014$. This means that an increase of only 100 nodes in the graph means that each node is included in 1.5 more clusters.

predictor	b-value	std. error	p-value
n	$1.47e^{-04}$	$2.73e^{-06}$	$< .001$
degree	$1.90e^{-01}$	$8.23e^{-04}$	$< .001$
bWaxman	$-4.33e^{-01}$	$4.17e^{-03}$	$< .001$
bTS	$7.74e^{-03}$	$4.97e^{-03}$	.12
bGrid	1.08	$8.52e^{-03}$	$< .001$
bExp	$2.11e^{-01}$	$5.29e^{-03}$	$< .001$

Figure 12: Regression table for cluster overlap

Node degree is not clearly a good predictor of cluster overlap ($b = 0.190$). Overlap depends largely on the structure of the graph rather than simply on the

number of connections. This will be evident in the coefficients for the various topology types.

As before, the random topology is the reference category. Being a grid topology had a clear influence on overlap with $b = 1.08$. This means simply being a grid topology meant that nodes were, on average, in 1.3 more clusters than they would have been had the topology a random topology with similar characteristics.

Neither Waxman topologies ($b = -0.433$) or exponential topologies ($b = 0.211$) had a clear increase over random topologies.

The difference between the Transit-Stub topologies and the random topologies was not statistically significant ($p = 0.12$). Since the transit stub topology clustered mostly cleanly of the various topology types due to its hierarchical nature, it naturally had very little effect on overlap.

4.2.4 Run-time Analysis

Refer back to figure 1 for the clustering algorithm. At the top level, the clustering algorithm runs once for each vertex. The second level, at which the algorithm runs through each cluster neighbor until it cannot add more vertices, is tricky to determine. We present a worst-case run-time based on a fully connected graph.

In the worst-case, a fully connected graph, every vertex is eventually added to the cluster. As a cluster is grown from a particular vertex, the SCM value creating by adding every other vertex to the cluster must be calculated before

any vertex is picked to be added to the cluster. After each vertex is added, this process must be repeated because the SCM value will be different for each node added. While there may be optimizations to the SCM calculation, these are effectively modifiers to the constant part of the $O()$ equation.

Here we examine the node addition step. Let n be the size of the set of vertices. For the i-th iteration in the worst-case, $n-i$ vertices must be tested for addition to the cluster, because there are n neighbors of the cluster minus the i existing cluster members. This is $\sum_{i=0}^{n} n-i$. To simplify we will call this $O(n^2)$.

So, in the centralized form, the algorithm will run in $O(n^3)$ time in the worst-case, that is, $O(n^2)$ run for every vertex. In the decentralized form, each node will perform the clustering in $O(n^2)$ time.

There are some additional limits that can be imposed on the running time. First, for the i-th iteration, the number of nodes tested is less than the current cluster size times the average vertex degree. In the worst case, this is close to n. In a sparse graph, the average degree is significantly less. Additionally, some techniques could be used for pre-culling some vertices so that they don't have to perform the clustering at all. For instance, only one of two adjacent vertices which would include each other in their clusters need build the clusters. This would have to be investigated to see if it had an effect on the clustering accuracy.

It is important to note that without modification, the worst-case running time of the algorithm in centralized form is $O(n^3)$.

4.2.5 Clustering Comparison

Next we will compare the clustering accuracy of SACA and SCSN. We use the average SCM value across the graph to represent the clustering accuracy.

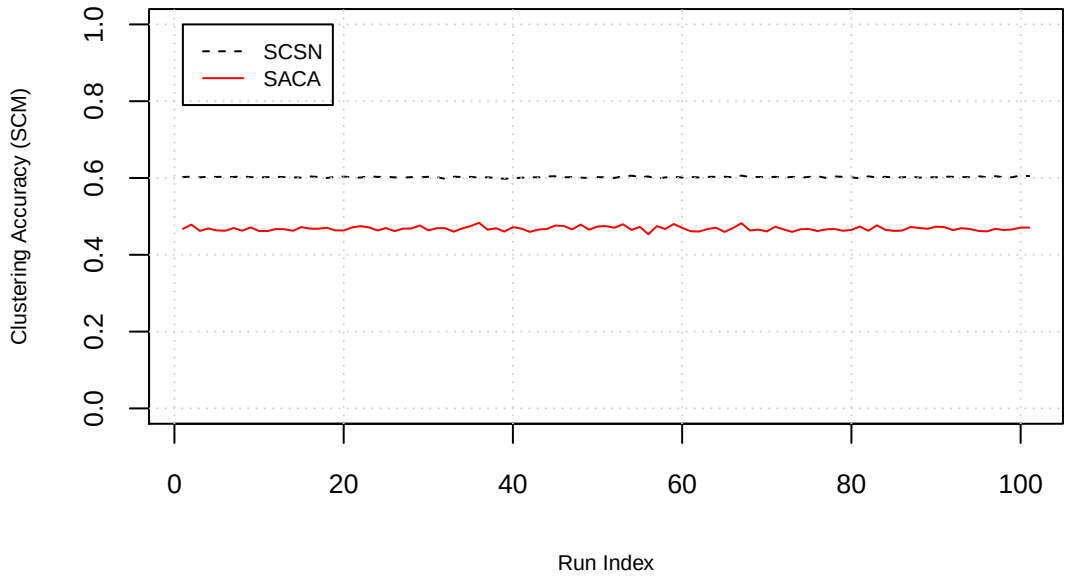

Figure 13: SCM variation on a 1000 node graph

Figure 13 shows the variation in SCM on a 1000 node graph over repeated runs of both SACA and SCSN. The graph is a Transit-Stub graph generated with 1000 nodes. Both algorithms were run repeatedly and the node SCM values were calculated and averaged for the entire graph. The mean SCM value for SACA is 0.47 with a standard deviation of 0.56e-2. The mean SCM value for SCSN is 0.60 with a standard deviation of 0.16e-2.

	mean diff.	t-value	df	p-value
SCM	0.13	1270.9	19776.7	$< .001$
Cluster Size	1.24	1156.5	13853.6	$< .001$

Figure 14: Mean difference of SCM and Cluster Size between SACA and SCSN

Running a t-test on the SCM values produced by the SCSN and SACA algorithms produces a statistically significant mean difference of 0.13. This means that when run on the same graph, SCSN clustered more accurately, with an value 0.13 higher than SACA. Figure 14 has the details.

The mean difference of cluster size is similar for SCSN and SACA. In this graph, SCSN clusters contained on average 1.24 more nodes than SACA clusters. Again see figure 14 for the results of the t-tests. As noted by the figure, the difference is statistically significant.

Speaking practically, SCSN can produce a more accurate cluster, reflected by the higher SCM values, by accessing additional nodes for each cluster. The resulting clusters better reflect the true cluster structures in the graph.

Figure 15 shows the SCM variation on a series of graphs varying in size from 1000 nodes to 8000 nodes. The average node degree remains similar in all of the graphs. The mean SCM value for SACA is 0.48, and the mean SCM value for SCSN is 0.60. While there is little variation in the SCM value as the number of nodes increases, SCSN reliably produces a higher clustering accuracy as measured by SCM value.

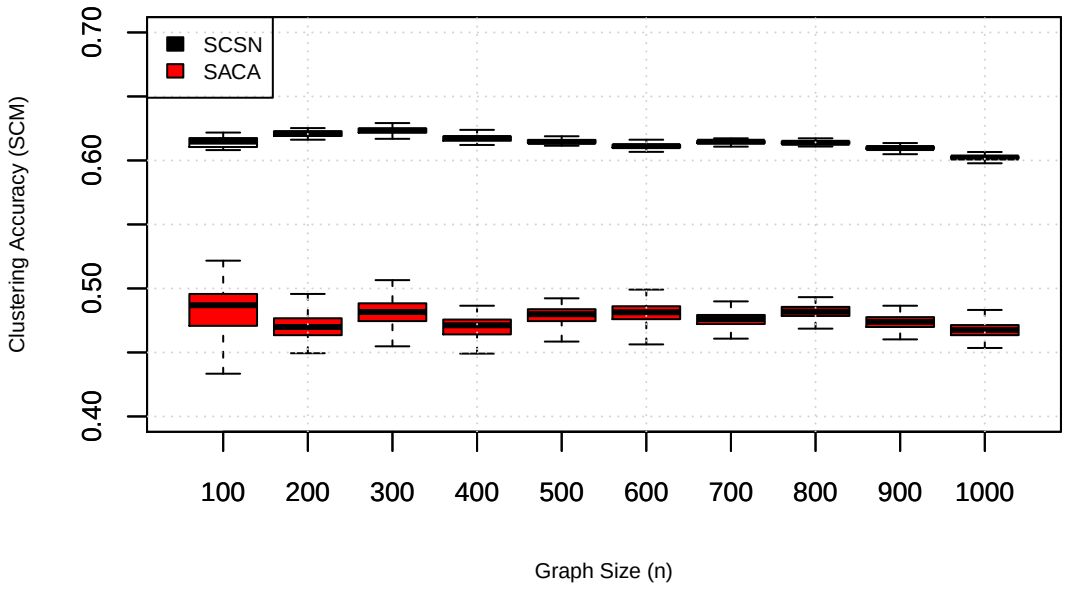

Figure 15: SCM variation on various graph sizes - Transit Stub

Figure 16 is similar to Figure 15 but it shows the SCM variation on a series of Exponential networks. Despite the radical difference in network configuration, we see the same pattern emerge. The SCM values are higher as produced by SCSN. SCSN is better able to cover all nodes in the graph, by allowing them to cluster with previously clustered nodes.

SCM measures node coverage, which SCSN is clearly able to optimize for. The question of weather the clusters are meaningful is not one that SCM can identify. We will attempt to answer that question next, as we talk about group identification.

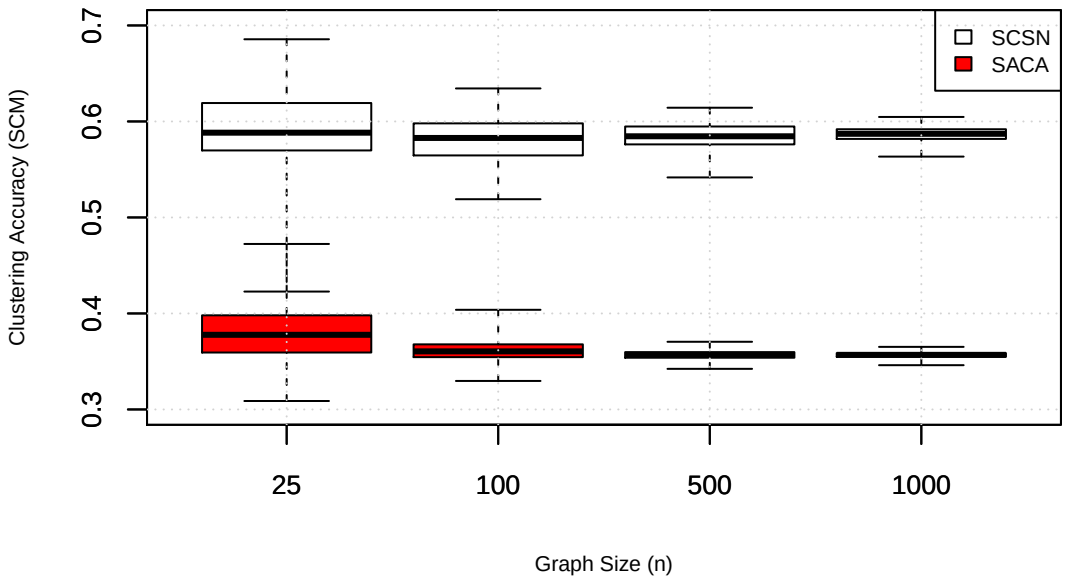

Figure 16: SCM variation on various graph sizes - Exponential

4.2.6 Group Identification

Having discussed the clustering algorithm's characteristics in the previous section, we will now examine its ability to identify groups from topological information. Our primary metric is the Average Graph Edit Distance (AGED), which was previously discussed. Recall that the AGED is an basically an average of the normalized Set Edit Distance, which is the percent of wrong nodes in a set. As a frame of reference, the AGED can be thought of as the average percent of wrong nodes for the clusters in the graph.

Lacking available overlapped clustering algorithms, we have chosen to use the SACA clustering algorithm as a reference point for our group identification.

Recall that the link density in our graphs is related to a link probability (P), which prunes random links between nodes. In our graphs in which P is high (close to 1), it would make sense that it would be easier to identify groups. Consider figure 17. When P is 1, SCSN matches the groups perfectly. As P decreases, the matches get worse. As expected, SACA is unable to match the groups perfectly since it is unable to handle overlapping clusters.

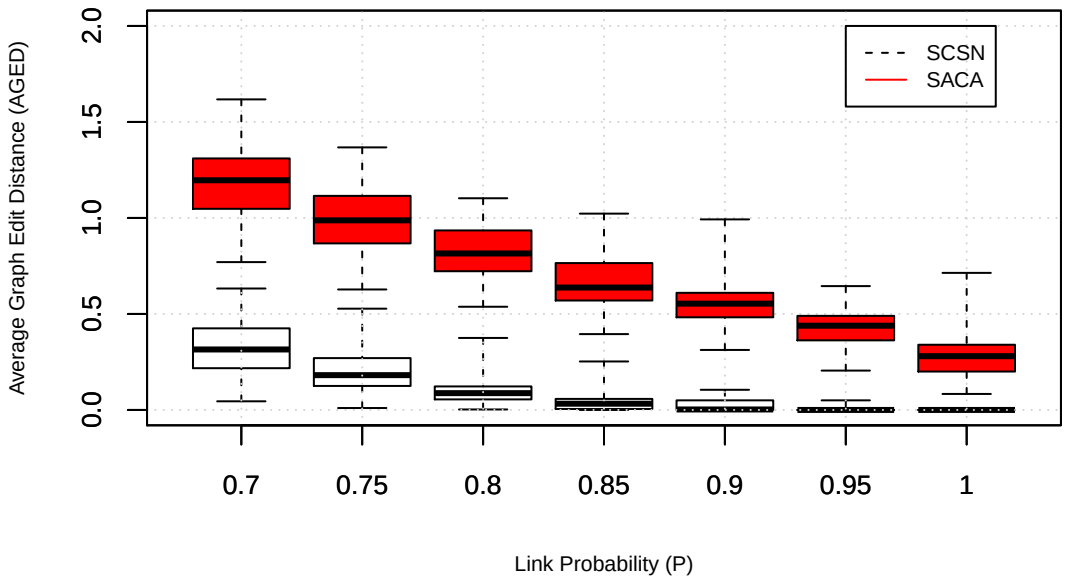

Figure 17: AGED variation on graphs of various link densities

The AGED metric includes missing and extra clusters as identified by the clustering algorithms. Interestingly, SCSN, while detecting overlapping clusters, tends to also identify clusters that were not originally in the group list, while SACA, unable to identify overlapping clusters, tends to lose clusters entirely,

presumably collecting two overlapped clusters into one. Figure 18 is the same as figure 17 except that it does not include missing and extra clusters. Note that the AGED for both algorithms decreases greatly (and note the change in scale on the vertical axis). Without accounting for missing and extra clusters, the AGED ranges from (0, 1) inclusive. The patterns from figure 17 hold here, as well. Note how the slopes have decreased, indicating that both algorithms suffer from missed and extra clusters.

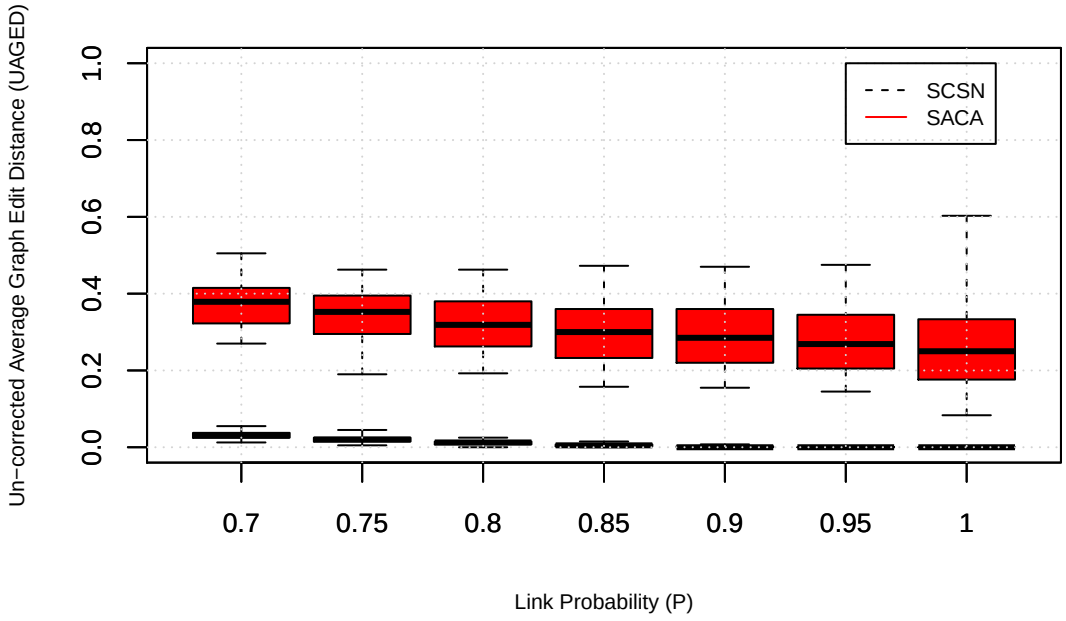

Figure 18: AGED variation on graphs of various link densities. AGED here is not corrected for missing and extra clusters.

Next, let us look at how the algorithms handle the groups which they identify correctly. For this, we will continue to look at the UAGED, that is, the Un-corrected AGED, which doesn't take into account missing or extra clusters.

Figure 19 shows the UAGED for a variety of degrees of overlap in various graphs. Note that the amount of overlap does not affect the accuracy of identifying the clusters in any discernable way, and that the clusters are identified quite accurately.

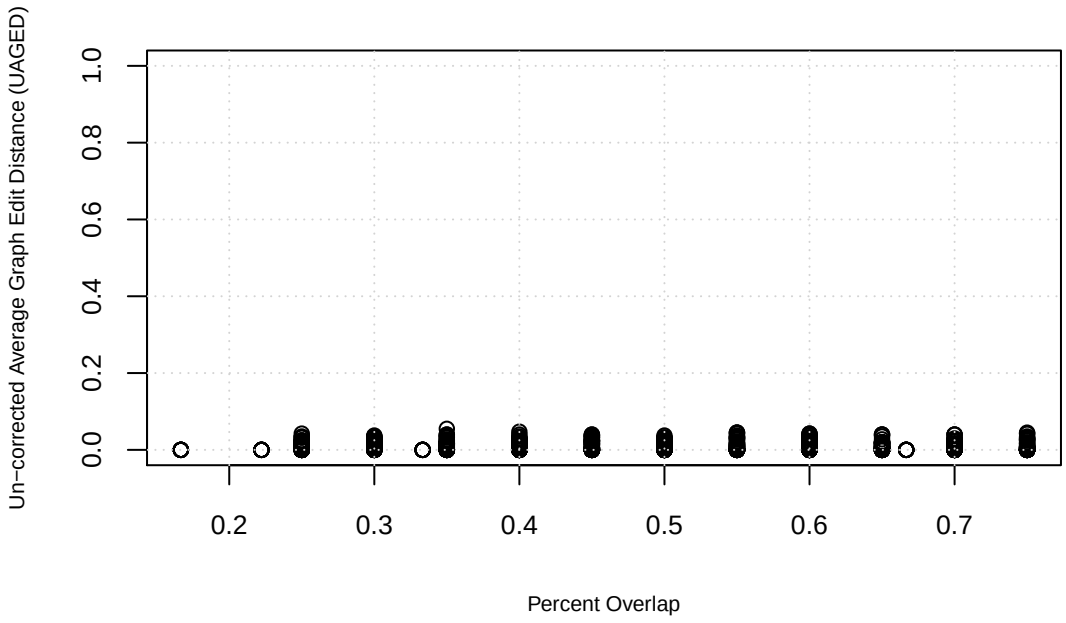

Figure 19: UAGED plotted against node overlap for SCSN

Figure 20 shows the same graph for the SACA clustering algorithm. As you would expect, SACA gets predictably worse as the overlap increases because it is not capable of identifying overlapped clusters. Note also the wider range of edit distance produced by the SACA algorithm. Recall that for UAGED, a value of 0.2 means that in a ten node cluster (on average) 2 nodes will be wrong in some

way. This could be two extra nodes, two missing nodes, or two nodes incorrectly identified as members of the cluster, or some combination of these.

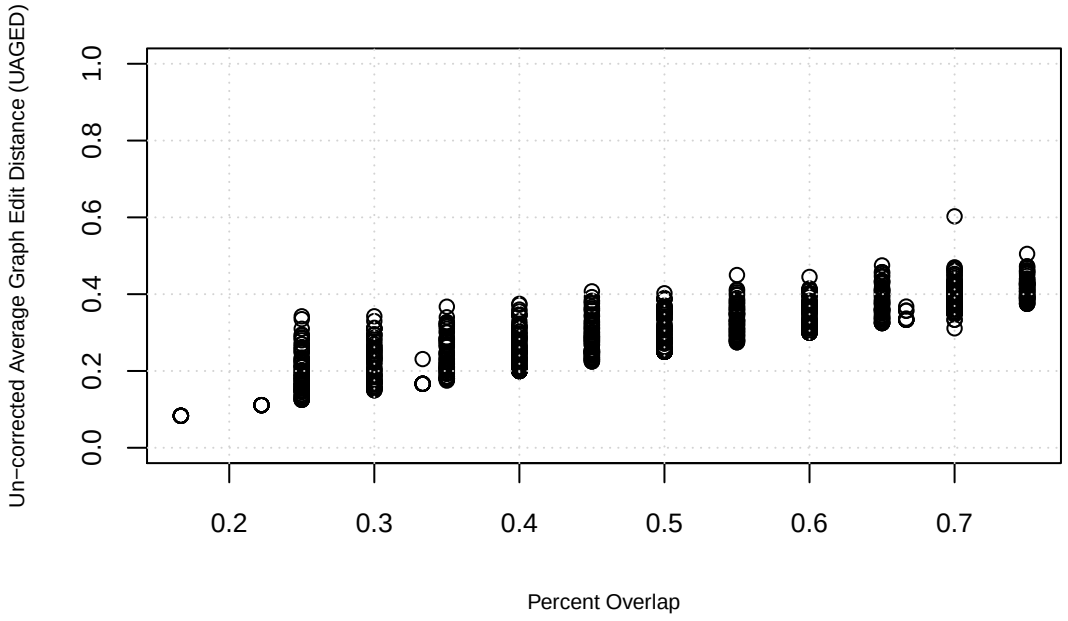

Figure 20: UAGED plotted against node overlap for SACA

Finally, we will present a few points from our data that are worth noting. On the same graphs, SACA and SCSN produce greatly different AGED values. Note that across all of our data, SCSN produces mean AGED less than 0.1, which implies that in this data set at least, less than one node in a ten node cluster is wrong, taking into account missing or extra clusters. Figure 21 provides a summary overview of the various graph edit distances.

On the subject of missing and extra clusters, SCSN produced no missing clusters, and a maximum of 12 extra clusters. The 12 extra clusters came in a

	SACA		SCSN	
	UAGED	AGED	UAGED	AGED
1st. Quartile	0.24	0.39	0.00	0.00
Median	0.31	0.61	0.00	0.01
Mean	0.30	0.66	0.01	0.09
3rd Quartile	0.37	0.91	0.02	0.13

Figure 21: AGED and UAGED in the data

graph of 20 clusters, so this is clearly a problem. SACA produced a few missing clusters (min. 2) and found a maximum of 25 extra clusters, also in a 20 cluster graph.

4.3 Routing Algorithm

Next we will present the results of our simulations on the routing algorithm. First we will present some descriptive statistics of the routing algorithm as it uses the SCSN clustering algorithm to perform the clustering. Next, we will present a comparison of the routing algorithm using both the SCSN clustering algorithm and the SACA clustering algorithm.

4.3.1 Cluster-Heads

The routing algorithm selects a number of cluster heads for operation. The ideal case is achieved in the first iteration of the routing algorithm, when the best possible nodes can be used for routing.

Figure 22 presents regression analysis results for the number of cluster heads elected in the first round of the routing algorithm. The adjusted R^2 is 0.71 for

this regression. The number of nodes (n) and the average node degree (*degree*) are multi-collinear but n is not very predictive on its own. The same is true for the mean size of the clusters (*SCSNcsz*) and the mean overlap of the clusters (*overlap*), where mean cluster size is not predictive on its own. As a result, number of nodes and mean cluster size were dropped from the model, which resulted in an improved adjusted R^2.

The clustering accuracy (*SCSNscm*) was a good predictor of the number of cluster heads ($b = 51.30$). Since the SCM value measures the quality of the clustering and the number of cluster heads is dependent on the clustering, it follows that SCM should be a good predictor. It is interesting to note that an increase in SCM value results in an increase in the number of cluster heads. Presumably, this is because as the clusters become larger, the SCM value will tend to decrease. When the clustering algorithm grows clusters, the SCM value will increase (as the clusters increase) as the algorithm adds nodes, because it is capturing more of the natural clustering features of the graph in the clusters. However, in the routing algorithm, a lower SCM could result in less cluster heads because the clusters can have a larger diameter. This would mean that SCM would go down due to nodes that are not as well connected, but the number of cluster heads would go down because a single cluster head could cover more nodes.

The mean number of clusters per node (*Overlap*) is also a good predictor of the number of cluster heads ($b = -1.16$). As more clusters overlap, that is, nodes

are in more clusters, less cluster heads will be required to cover all the nodes. Node that mean clusters per node is a negative predictor of the number of cluster heads. More clusters per node means less cluster heads.

predictor	b-value	std. error	p-value
SCSNscm	51.30	4.33	$< .001$
Overlap	-1.16	0.40	$< .005$
nclust	0.16	0.01	$< .001$
degree	-0.44	0.09	$< .001$

Figure 22: Regression table for the number of cluster heads

The number of clusters produced by the clustering algorithm(*nclust*) is not as useful of a predictor ($b = 0.16$). As the number of clusters increases the number of cluster heads should also increase. However, because of the effects of overlap, this is lost, even with the linear regression model controlling for overlap.

Mean node degree(*degree*) is slightly more useful as a predictor ($b = -0.44$). Mean degree is also captured to some extent by the clustering features, covered by SCM.

4.3.2 Control Messages

Control messages are sent to set up the routing tables used by the routing algorithm. Naturally, there is a strong correlation between the number of nodes in the graph and the number of control messages sent ($r = 0.98$). The regression model presented in Figure 23 shows some additional relationships to the number

of control messages required. Note that adjusted $R^2 = 0.98$ and all p-values are less than .001.

Interestingly, the number of nodes in the graph and the average node degree (*degree*) again seem to be multi-collinear and degree does not seem to be strongly predictive. As a result, average node degree was dropped from the regression. Mean cluster size was dropped for the same reason as in the analysis of the number of cluster heads.

Clustering accuracy is strongly negatively predictive of the number of control messages. Since control messages are only necessary within clusters, this makes sense. If the cluster is multi-hop (resulting in a lower SCM value) more control messages are required to be sent to cover the entire cluster.

The mean number of clusters per node (overlap) is predictive of the number of control messages ($b = 16.76$). The more overlap, the more nodes are covered by a particular cluster head. The model indicates that for every additional cluster per node, there are 16 more control messages sent. Number of clusters is again a weak predictor.

predictor	b-value	std. error	p-value
scm	-95.00	26.26	$< .001$
overlap	16.76	1.34	$< .001$
nclust	-0.58	0.10	$< .001$
n	2.51	0.04	$< .001$

Figure 23: Regression table for the number of control messages

The number of nodes is a strong predictor of the number of control messages. In this model and data set, ($b = 2.51$). That means that regardless of degree (inclusion in the model does not significantly change this b value) and controlling for the other predictors, each node is responsible for 2.5 control messages.

4.3.3 Data Messages

Finally we will model the predictors of number of messages required to collect all the data at the cluster heads. In this linear regression model, shown in Figure 24, the adjusted R^2 value is 0.96. The predictors not included in this model are the same as for the control messages model.

Clustering accuracy is an excellent predictor of the number of data messages required ($b = -101.72$). The primary goal of the routing algorithm is to reduce the number of data messages required by leveraging the clustering algorithm, so this is a good indicator that this goal is being met.

Mean number of clusters per node (overlap) is a strong predictor of the number of data messages ($b = 15.31$). Number of clusters is not ($b = -0.5$). This is similar to the discussion for the number of control messages.

predictor	b-value	std. error	p-value
scm	-101.72	29.50	$< .001$
overlap	15.31	1.51	$< .001$
nclust	-0.50	0.11	$< .001$
n	1.52	0.05	$< .001$

Figure 24: Regression table for the number of data messages

The number of nodes in the graph is not as strongly predictive as in the regression model for control messages. Every node in the graph is still sending data messages ($b = 1.52$).

4.3.4 Routing Performance

4.3.4.1 SCBR Overlapped Cluster Performance Analysis

Here we will discuss the performance of the clustering algorithm using both overlapped (SCSN) and non-overlapped (SACA) clustering algorithms.

To compare the routing performance of the clustering algorithms we consider primarily the number of cluster-heads discovered by the routing algorithm under each circumstance. We will normalize the number of cluster-heads to the size of the graph and compare the mean number of cluster-heads. The analysis is performed on the same graphs for both clustering algorithms.

Figure 25 shows the normalized number of cluster-heads between SACA and SCSN as discovered by the SCBR routing algorithm.

	SCSN Cluster-heads	SACA Cluster-heads
1st. Quartile	0.025	0.15
Median	0.04	0.20
3rd. Quartile	0.10	0.29

Figure 25: Normalized cluster-heads between SACA and SCSN clusters

Figure 26 shows the mean difference between the number of cluster-heads (normalized) between the outcomes of the SCSN and SACA routing algorithms.

The SACA algorithm produces a mean of 0.224 cluster-heads per node and the SCSN algorithm produces a mean of 0.066 cluster-heads per node. The mean difference is 0.16 and is statistically significantly different from zero.

	mean diff.	t-value	df	p-value
Cluster-heads	0.16	43.6	1363.7	$< .001$

Figure 26: Mean difference of SCBR Cluster-heads between SACA and SCSN

The benefit of SCSN's overlapping clustering is that more nodes can be covered by less cluster-heads. Regardless of the number of clusters that are produced, because nodes can exist in multiple clusters, cluster-heads can cover more nodes than in the SACA routing algorithm.

4.3.4.2 SCSN Performance

The routing algorithm is based on the idea that short transmissions are better than long transmissions. In the naive case, every node must make a long distance transmission in order to communicate data to the base station. In the case of our routing algorithm, only the cluster heads must make transmissions to the base station. We will attempt to determine the power difference required to make a savings in radio transmissions.

Figure 27 shows the two types of routing messages and their medians in our simulation data. When you break the data down, these values remain similar. The values are normalized to the number of nodes in the graph, because the number of messages is directly related to the number of nodes sending messages.

	n-cm	n-dm
1st. Quartile	2.05	1.12
Median	2.29	1.33
3rd. Quartile	2.39	1.43

Figure 27: Normalized Control Messages and Data Messages

Figure 27 tells us that in the median case, the routing algorithm requires 229 control messages per hundred nodes, and 133 data messages. That is, therefore, 362 messages per hundred nodes, or 3.5 messages per node. If the elections are only performed every few rounds of messaging, that goes down further. We can conservatively say that if the long distance messaging power cost is around 4 times the cost of the short transmission power cost, we have achieved savings. If the data messages can be transmitted four times before new routing elections happen, it improves our total messaging to 190 messages per hundred nodes, and we only need the high power cost to be double before we achieve savings

Chapter 5

Conclusion

5.1 Clustering Algorithm

This paper presents, first and foremost, an algorithm for topological graph clustering. The clustering algorithm performs well as far as building quality clusters with good node coverage. The node coverage of the various graphs tested, as evidenced by SCM values, is better than that of the SACA clustering algorithm. While SACA may find nodes that can't reasonably be clustered without penalizing other nodes, in the SCSN algorithm, every node belongs to at least one cluster.

It is vulnerable to detecting excessive numbers of clusters because every possible combination of nodes is a potential cluster, excluding subsets, which are discarded. Consider three nodes, A, B, and C. Regardless of their connectivity, they can be clustered into two node clusters AB, AC, BC. It is not possible for a

non-overlapped clustering algorithm to create these clusters simultaneously, but it is possible for SCSN. Despite these unique challenges, we feel that we have presented data that the clustering provides a meaningful advantage in identifications of overlapping groups.

Because the SCSN clustering algorithm is not affected by encountering adjacent clusters, it suffers a penalty in running time as compared to SACA. In the worst case, the running time is prohibitive, but in a sparse network, the running time may be such that the clustering algorithm is still useful. In addition, because the local connectivity information is all that is needed, SCSN is scalable as long as the local network information can be delivered to the nodes in the network in a reasonable time. An algorithm for distributing the node connectivity information is beyond the scope of this work.

The most important aspect of the clustering algorithm is the quality of the clusters that it detects. In this paper, we have shown that the overlapped clusters detected by the algorithm are close to the original groups from which the graphs were generated. Lacking an empirical standard for comparison, we have shown that the clustering is at least much improved from an algorithm that cannot identify overlapped clusters, even when the amount of overlap is small. When the overlap is large, the quality of the clustering remains high with the SCSN algorithm.

Additionally, This work presents a metric that can be used for the evaluation of group identification clustering algorithms based on Levenshtien distance, also

known as edit distance. This seems to be a useful metric, as it is not only widely used in other applications, but is relatively simple and gives a realistic sense of what it is measuring. Future works can compare to SCSN using this metric.

Finally, the overlapped clustering provides the basis for our hierarchical routing application.

5.2 Routing Algorithm

The routing algorithm takes advantaged of overlapped clustering to send data in a sensor type network. Several of the clustering characteristics seem to be predictive of the routing characteristics and we have attempted to determine what aspects of the clustering best contribute to the quality of the routing.

The routing algorithm clearly benefits from using overlapped clusters to identify cluster-heads, as demonstrated by a reduced number of cluster-heads on the same graphs. The facilitates better coverage of the graph with less nodes.

We have also demonstrated that given a fairly simple set of constraints, the routing algorithm can produce a power savings for fixed position sensor networks by leveraging hierarchical routing techniques and the clusters identified by the clustering algorithm. This routing algorithm may be useful in specialized sensor networks, where more generalized algorithms may not be applicable.

5.3 Future Work

Here we will present some future work that grew out of the development of this work and open issues that we have encountered.

5.3.1 Clustering

It should be possible to create an SCM measure that takes into account the distance of a neighbor when calculating the SCM of the vertex. So if a node is clustered to a single hop neighbor, the node will get more benefit than if it is clustered to a multiple hop neighbor. Additionally, it could be useful to create an SCM type metric that accounts for edge weights.

Overlapped clustering could be applied to topology generation to attempt to generate realistic network topologies. A network topology could be connected as a series of clusters or the probability of a link could be influenced by the clustering.

The SCSN algorithm's ability to reproduce quality input clusters needs to be investigated on real-world graphs. Social network graphs are possible candidates, having group structures that can be identified through meta-data, which will provide good comparison data for the quality of the clustering algorithm. While the SCSN algorithm appears to do a good job identifying pre-existing groups in randomly generated graphs, the patterns of overlap in real world data may make it more (or less) difficult to identify pre-existing groups. Additionally, investigation of real-world graphs will make it possible to build better models of and generate

better test graphs. As social networks become more important, algorithms for examining group movement will also be more important, and having quality test data will be useful, especially in the absence of freely available social network data.

5.3.2 Routing

The ideal test for the routing algorithm short of a true implementation would be an implementation on a network simulator, which provides the underlying network and would enable message counting in a more realistic condition. However, such a simulator would need to be able to handle the conditions for which we developed the routing algorithm.

While related to clustering, a repair mechanism is needed to handle node movement once the clusters are built. A mechanism similar to the SACA algorithm's repair mechanism should be feasible, but would need to be tested against a fresh clustering to see if it would produce reasonable results. This would allow routing to handle limited node movement, assuming that the clustering updates were not too expensive a full clustering was needed only infrequently.

Since we are only considering fixed location networks, we have not considered the cost of clustering the network, because this is only done once. However, it would be useful to know the full cost of the clustering, and if expensive, this could be optimized.

It should be possible to build a backbone network through the overlapped areas of the clustering. This could be used as a more traditional routing algorithm to see if we can route through an ad-hoc network. Applying this as a pro-active routing algorithm for mobile networks would require that the clustering be repairable at a reasonable cost.

Bibliography

[1] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2(1):1 – 22, 2004.

[2] C.-C. Chiang and M. Gerla. Routing and multicast in multihop, mobile wireless networks. In *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, volume 2, pages 546 –551 vol.2, Oct. 1997.

[3] C. chuan Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. 1997.

[4] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. RFC 2501, Jan. 1999.

[5] S. V. Dongen. A new cluster algorithm for graphs. Technical report, National Research Institute for Mathematics and Computer Science in the, 1998.

[6] S. V. Dongen. Performance criteria for graph clustering and markov cluster experiments. Technical report, Amsterdam, The Netherlands, The Netherlands, 2000.

[7] M. Gerla, X. Hong, and G. Pei. Landmark routing for large ad hoc wireless networks. In *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, volume 3, pages 1702 –1706 vol.3, 2000.

[8] Z. J. Haas and M. R. Pearlman. The performance of query control schemes for the zone routing protocol. *IEEE/ACM Trans. Netw.*, 9:427–438, August 2001.

[9] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *Network, IEEE*, 16(4):11 –21, Jul./Aug. 2002.

[10] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks, 1996.

[11] Y. Li, J.-H. Cui, D. Maggiorini, and M. Faloutsos. Characterizing and modelling clustering features in as-level internet topology. 2007.

[12] Y. Li, L. Lao, and J.-H. Cui. Saca: Scm-based adaptive clustering algorithm. *Proceedings of IEEE MASCOTS*, September 2005.

[13] G. Pei, M. Gerla, and T.-W. Chen. Fisheye state routing: a routing scheme for ad hoc wireless networks. In *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, volume 1, pages 70 –74 vol.1, 2000.

[14] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A wireless hierarchical routing protocol with group mobility. In *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, pages 1538 –1542 vol.3, September 1999.

[15] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24:234–244, October 1994.

[16] E. M. Royer, C. E. Perkins, and S. R. Das. Ad hoc on-demand distance vector (aodv) routing, 2000.

[17] D. Waitzman. Standard for the transmission of ip datagrams on avian carriers. RFC 1149, April 1990.